MANAGING CONFIGURATION OF GROUND SOFTWARE APPLICATIONS WITH GLUEWARE

Barbara Larsen
Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
Barbara.S.Larsen@jpl.nasa.gov

Randy Herrera
Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
Randy.G.Herrera@jpl.nasa.gov

Tadas Sesplaukis
Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
Tadas.Sesplaukis@jpl.nasa.gov

Leo Cheng
Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
Leo.Y.Cheng @jpl.nasa.gov

Marc Sarrel
Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
Marc.A.Sarrel@jpl.nasa.gov

Abstract

Managing the specification of input data in the uplink subsystem has always been nontrivial—multiple applications must use a correct and consistent file set that changes as sequence development progresses in time. It is, however, important since misconfiguration introduces risk and requires rework. The distributed operations environment of the Cassini-Huygens project provides additional challenges with remote users and remote machines, multiple system architectures, and fewer specialized operators. Because the lower cost paradigm for mission operations was instituted after the mission concept was in place, addressing this complexity in the ground system has been problematic. The operations team thus faces the conundrum of having neither the resources for business as usual nor any significant funding for simplification.

This paper reports on a simple, low-cost effort to streamline the configuration of the uplink software tools. Even though the existing ground system consisted of JPL and custom Cassini software rather than COTS, we chose a glueware approach--reintegrating with wrappers and bridges and adding minimal new functionality. Highlights of the restructured system include the following: (a) an electronic version of the master list of correct ancillary files updated at each stage of sequence development, (b) a script validating the master ancillary list construction and verifying that named files exist, (c) a translator to make a web page reference from the master list, (d) a new tool which enables single point construction for all applications of sequence specific configuration files based on the master list of correct files, (e) alterations to the configuration files themselves and to already existing application wrappers so that those sequence specific specifications can be used, (f) a configuration file naming convention that allows easy recognition of the appropriate configuration file for the work at hand, (g) expansion of the project database to include the master ancillary file lists and the configuration files, (h) a logical file structure that provides the application programs a single view of the ancillary files while allowing different implementations in various subsystem architectures, and (i) active maintenance of the ancillary input files within the ground system in a manner expected by the applications.

Available resources for development demanded a solution that was inexpensive and evolutionary. By rethinking procedures, making modest changes to existing components of the ground system, and adding glueware, the configuration of uplink software applications was simplified and made more consistent. Cost savings result from eliminating redundant effort, increasing efficiency with simple automation, reducing risk, and saving disk space and bandwidth.

1. Evolution of the Configuration Process

During the cruise phase of the Cassini mission, while there was limited science activity and little overlap in development of different sequences, support for configuration of the sequencing software was minimal. The lead engineer for the sequence, in conjunction with mission planners, determined the correct ancillary files to use and published this list (in pdf format) on the web site for the particular sequence. Each user acquired the necessary files, mostly

from the project database with occasional out of the ordinary exceptions, and constructed configuration files as needed. While there was no compelling need to alter this approach, weaknesses did appear. Users complained that the applications were hard to initialize. Some were uncomfortable dealing with configuration files at all. With users operating on multiple systems at both local and remote sites, obtaining the files was occasionally problematic. Rare errors of configuration were observed. Also, the process was inefficient. The same tasks of obtaining the files and constructing the configuration file had to be done independently by each user. At this point in the mission, though, the savings from centralizing and automating these tasks could not offset the costs of addressing the inefficiency.

When detailed development of the Cassini tour began in May 2002, the complexity of the configuration process took a sudden jump, with more users working to tighter timelines on more simultaneous sequences. Almost immediately, the lead engineers realized that a central configuration file was essential, although their concern was mostly to ensure use of correct files rather than to save time or effort. They began to hand construct sequence specific configuration files for the two most heavily used applications. While this represented a noticeable improvement for the user community, the work represented a burden to both the people tasked with constructing them and to the sequencing software system engineers for support of that construction. Also, dealing with each sequence and each application independently introduced complications. Each application required a separate process for constructing its configuration file, but the ancillary files specified within had to be consistent. Each sequence required a unique set of ancillary files, but with appreciable redundancy from one sequence to the next. In locally managed systems, this resulted in noticeable consumption of disk space by duplicate files.

As experience grew, the need for a system-engineered solution became obvious. However, the resources to achieve such a solution were negligible. A working group was convened to identify a new low-cost process for managing the ancillary files required by the sequence software system. Changes had to address the above concerns with minimal disruption of the larger processes into which they would be integrated.

To distribute the implementation work and allow for incremental development, we chose to use glueware. A handful of minor modifications to custom Cassini software and existing wrappers were augmented with wrappers and bridges in PERL and minimal new functionality.

2. Components of the Glueware System

Our first step was a fundamental but uncomplicated restructuring of the Cassini ground system, creating a new component to hold the ancillary files where they could be actively maintained with little effort and would be accessible to software users with minimal or no action on their part. A logical structure was defined so that application programs have a single view of the file system. The branches of this ancillary file tree are illustrated in Diagram 1.

Diagram 1. The Ancillary File Structure

NAIF
- ephemeris
- *leapseconds*
- *planetary constants*
- *frame kernel*
- *instrument kernel*
- *clock*

DSN
- *allocation*
- *viewperiod*

NAV
- *geometry*
- *lighttime*

SPACECRAFT OPS
- turn limits
- *clock*

UPLINK TEAM
- *initial conditions*

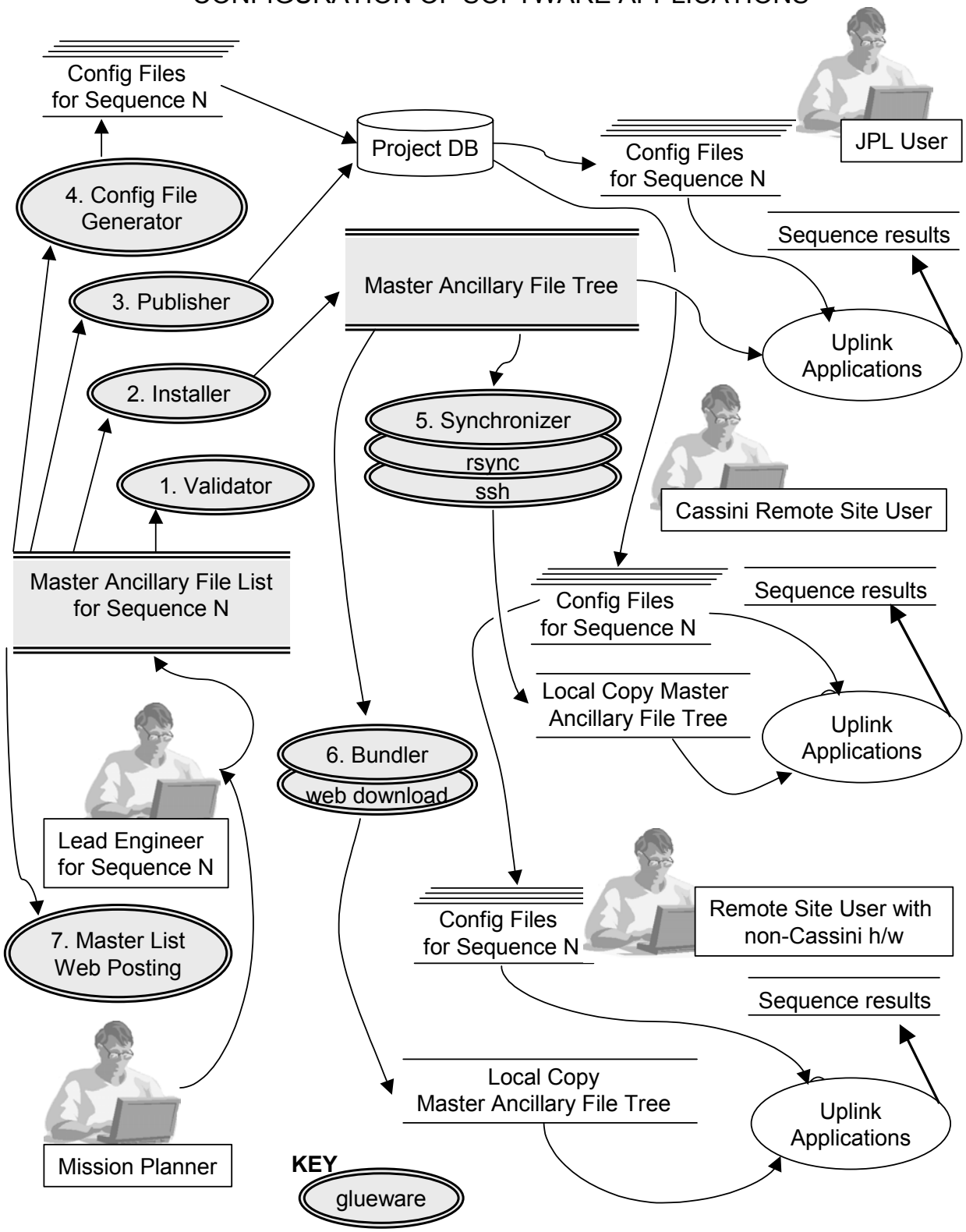
SCIENCE PLANNING
- epochs

Italicized files are links to project database at JPL, actual files at distributed sites.
Non-italicized are files everywhere.

Multiple physical implementations were required since Cassini does not have a single ground system architecture. A separate approach was required for the Cassini networks at JPL, for the approximately 25 machines at remote sites that are managed by JPL, and for the remote sites that are independent. The master file tree was created in JPL afs space. At each distributed site, the tree is locally replicated.

The on-going synchronization of the file trees on distributed Cassini hardware with the master tree at JPL proved challenging because there was no ownership of the new subsystem within operations and no development responsibility between deliveries of the software. To make the task easy enough that the lead engineer could include it with his responsibilities, another glueware piece was added. The synchronizer script (#5 in Diagram 2) is built over

DIAGRAM 2. PROCESS FOR MANAGING
CONFIGURATION OF SOFTWARE APPLICATIONS



rsync, an open source utility for fast incremental file transfer. The script loops through the list of machines to be synchronized, trying each three times to overcome transient network problems. Since the connection is established usingssh, and the ssh agent employs public key authorization, the authority of the script user to make the changes needs to be established only once per execution rather than once per machine. The script also sends email notification, once at start and again at finish with a summary of results.

Rsync was chosen because of several advantages over a simple secure copy, particularly when implemented with conservative choices in its options. Checksum verification can be required rather than relying simply on timestamp and file size. If files are the same, no action is taken. If files are different, only differences in the files are transmitted over the link. When a file is added to the master tree, it is also added at the remote site; likewise, when a file is deleted from the master tree, it is removed at the remote site. Rsync will make an actual copy of the file on the remote site when the file at the master site is an "unsafe" link, i.e. outside the directory subtree which is being synchronized, as is the case for many of the files in the master ancillary file tree. For initial installation, the total size of the files transferred was about 130 Mbytes. The total time for full transfers to all machines was about four hours and fifteen minutes. The variation across the distribution of Cassini sites is shown in the table below.

LOCATION	TRANSFER TIME
JPL	3 minutes
United States	10 to 15 minutes
Europe	25 minutes

If no files need to be transferred, each machine, regardless of location, takes about 30 to 45 seconds. Experience has born out the expectation that rsync will be a thorough and efficient choice for synchronizing the file trees at the remote sites.

For users with non-Cassini hardware, the file tree structure was included with the latest revision of the downloadable version of the software. Sequence specific updates are made available via the download website. These updates consist of a tar bundle of files that will eventually be created by one of the glueware pieces, the bundler script (#6 in Diagram 2). The bundling includes the actual files for those in the master tree where the file is actually a link to the database.

The next step was to enable the applications themselves and the existing wrappers that invoke them to employ the ancillary files. The goal was to encapsulate all data that could change from sequence to sequence, i.e. independent of software deliveries, in files in the ancillary system. The needed changes required were modest. Largely, these could be accomplished by changing the configuration files and templates to refer to the ancillary file system rather than to local files. For the Pointing Design Tool in some limited instances, however, the configuration file had previously contained the actual data, e.g. for rate and acceleration limits, rather than referencing a file with that data. The capability was therefore added to specify a file within the configuration file that the reader would open and process as though that data had been within the configuration file itself. For the Science Opportunity Analyzer, the configuration file provides both application load instructions and user file selections. Here it was necessary to divide the configuration file into the items that are sequence specific and therefore refer to the ancillary files and into the load instructions. Although this work is not completed, the configuration file will be melded of these two pieces by the wrapper runscript that invokes the application. A similar separation into a data file of the sequence specific items for seq_gen, the sequence generator application, had previously been undertaken, so the only change for the ancillary file system was in the path specification in the data file.

To this point, the file system had been established and the software prepared to take advantage of it. The final task was to ensure that configuration files were available for the sequencing applications that specified a correct, consistent set of files. Step one was to define a configuration file naming convention making obvious the connection of the file to the port, phase, and sequence to which it applied. Next was the conversion of the master list of ancillary files, which had previously been a paper product, into an electronic list. The file types were assigned keywords so that the association between the master list and configuration file would not be ambiguous and so that the master list would have a known, parsable structure. Automation of the master list was ruled out because of the decision making involved; however, some potential software checks on the consistency of the file set have been identified and may be added to future versions.

Another piece of glueware, the validator (#1 in Diagram 2), checks the master list for correctness and completeness, verifying that all keywords are present and not duplicated. It also confirms that the named files exist.

For files that are in the project database but not in the ancillary file tree, the installer (#2 in Diagram 2) creates a new link between the master ancillary file tree and the project database where possible. For those whose format is different in the ancillary system (e.g. binary ephemeris files) the installer copies the file from the project database, then converts to the appropriate format.

When the list is ready, Mission Planning is notified. The mission planners play a significant role in determining the

correct set of files to use. In addition to reviewing the master list, they will use a script of their own devising (#7 in Diagram 2) to translate the master list into a web posting for a second method of access.

The configuration file generator produces sequence specific configuration files for the uplink applications. It currently consists of a separate script for each application but will be consolidated into single point construction of the set of consistent configuration files. Each script uses the keywords in the master file list to transform a template for the application into a configuration file that specifies the files in the master list. The lead engineers have expressed appreciation for the elimination of formatting, syntax, and spelling errors that occurred in hand constructed files.

Storage and distribution of the master file list and the configuration files was accomplished by adding collections for these files to the project database. While not every user of the applications has access to the project database, each instrument team already had in place a mechanism for redistributing the configuration files to their members without access. In Diagram 2, this is illustrated by the transfer of config files from the Cassini remote site user to the remote site user without Cassini hardware.

3. Reduced Costs

The benefits of the new process for managing ancillary files are in two realms—timesavings and risk reduction. The savings from labor reduction accrued from consolidating tasks that had been repeated by each user and from developing simple automation for tedious tasks. Consolidation of the files also saved IT costs by slowing demand for disk space, a chronic problem in the Cassini environment. The synchronization effort saved bandwidth by minimizing data transferred and making the transfer once per machine independent of the number of users relying on that machine.

Using the old process labor expended was as follows: The preparation of the configuration files without the glueware took approximately a half-day each time. For local users, obtaining the specified files and modifying the configuration file for the local setup took on the order of .5 hour. For remote Cassini users, the time estimate for this task was slightly higher due to increased effort to obtain the files. For remote users without Cassini hardware, adjusting for local differences added at least fifty percent to the local estimate for setup. Additional time was spent by system engineers in the tool development organization on analysis and correction of configuration file errors.

A conservative estimate of the cost savings is 1.3 work years. For the rest of the mission, there will be a minimum of 111 iterations of this process (29 sequences remaining in advance planning plus update and final sequence development for each of the 41 tour sequences). Early evaluation suggests at least a two-hour savings in the configuration file generation. The savings by users would be 111 × the number of users, which varies from sequence to sequence. However, with twelve instruments, opnavs, and downlinks there are at least fourteen users who would save most of the half hour and twelve who would save most of the longer setup. The need for system engineering should vanish, which is fortunate since funding for that function does not last until the end of mission.

The risk reduction accrues from using software checks to ensure that the files specified are consistent across applications. With the new process, the single act an end user has to take to employ correct files is to obtain the configuration file labeled for the sequence on which he is working. The rewards of this design will increase as the number of simultaneous sequences in development increases to as many as five. The process also ensures that there is an accessible, historical record of the files that were specified at each step in the development of a sequence.

4. Future Enhancements

As noted above, development of some of the pieces described is still in progress. Also, further thought will be given to the following improvements:

1. Using data within the ancillary files to check for consistent sets.
2. Making the set of tools for construction of the configuration files and setup of the ancillary files more integrated and easier to use.
3. Providing ancillary file tree synchronization via rsync to users with non-Cassini hardware.

5. Acknowledgements

The development described in this publication was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

The authors wish to thank the following contributors to the new process:

- The other members of the working group.
- The system administrators and CM engineers who assisted in setting up the file space and creating links to it: Dave Coppedge, Nick Patel, and Sheila Chatterje.

- The Pointing Design Tool (PDT) developers, Jeff Boyer, Joel Reynolds, and Kevin Yau, who added the needed changes to their next delivery with no schedule relief.

- The lead engineer, Jennifer Long, who first exercised the glueware.

6. Conclusion

A new process for managing the configuration of uplink applications was implemented at low cost with glueware. As a result, the lead engineers have software support for specifying the desired configuration of the uplink applications. The end users spend less effort to configure the applications and have less opportunity to make mistakes in file specification. Cassini benefits with cost savings from labor and risk reductions.